IDRiM (2025) 15 (1) **ISSN: 2185-8322**
DOI10.5595/001c.134130

Journal of Integrated Disaster Risk Management

Regular Article

An Integrated Machine Learning and Deep Learning Framework for River-Based Flood Early Warning System

Leishangthem Sashikumar Singh[1], Ksh. Nilakanta Singh[1], Nazrul Hoque[1] and Khumukcham Robindro Singh[1*]

Received: 12/09/2024 / Accepted: 14/02/2025 / Published online: 17/04/2025

Abstract The impact of global climatic conditions and unpredictable rainfall patterns has led to numerous natural disasters in Manipur, including floods and landslides. Developing a reliable flood forecasting and early warning system is crucial to prevent loss of life, property, infrastructure, and crops. This paper presents a machine learning and deep learning framework for flood prediction, comprising four modules: Data Collection, Data Manipulation, Model Implementation, and Early Warning and Flood Forecasting. By leveraging daily data from gauge stations and weather reports, the framework aims to predict river water levels accurately, particularly during the monsoon season. The objectives include developing a model that integrates K-Nearest Neighbor (KNN) with Long Short-Term Memory (LSTM) networks for predicting future water levels using multivariate time series data and providing an algorithm that offers early warnings for potential floods. The proposed framework employs a multivariate time-series approach using a hybrid KNN-LSTM method to predict river water levels. Missing data are imputed using the KNN algorithm with historical weather data, and an LSTM network predicts future water levels based on this information. This approach effectively models temporal dependencies across multiple gauge stations in the Imphal Valley, Manipur. The performance metrics for the proposed model are NSE score of 0.9393, MAPE score of 0.1076, and RMSE score of 1.1828, which are relatively better than those of the conventional approaches. This study presents a novel flood forecasting approach by integrating for imputing missing data with LSTM networks to capture temporal dependencies. The framework is applied to homogeneous regions with correlated time series, effectively modeling the complex dynamics of river water levels across multiple gauge stations.

Keywords: flood prediction, water level, rainfall pattern, machine learning, artificial neural networks, deep learning

[1] Department of Computer Science, Manipur University, India
* Corresponding author email: rbkh@manipuruniv.ac.in

1. INTRODUCTION

Floods are among the most widespread natural disasters worldwide. Although once uncommon in Manipur, they are now increasingly frequent in the Imphal valley during the rainy season. Manipur, a small state located in the northeastern part of India, faces several primary causes of flooding in the valley such as increased urbanization, changes in land use patterns, high-intensity and unpredictable rainfall, heavy runoff, and low infiltration rates. Additionally, the lack of early warning systems and long-term planning exacerbates flood-related problems in the region. In recent years, Manipur's rainfall patterns have changed, leading to more frequent floods. This has significantly impacted the socio-economic conditions of the majority of farmers, as agriculture is the primary occupation in Manipur. Developing a reliable and accurate flood prediction model is essential for sustainable flood risk management (Adam, 2020), focusing on prevention, protection, and preparedness (Ridzuan et al., 2024).

Estimating hazards and managing the risks of extreme events are essential when implementing a flood forecasting model. A robust and accurate prediction model is crucial for disaster management, as it analyzes data, informs policy decisions, and guides evacuation strategies. Spatial and temporal data related to river water levels are vital for precise predictions. Lead time prediction models, including short-term and long-term flood predictions, are actively emphasized to mitigate the loss of lives, property, and the environment (Mosavi et al., 2018). Moreover, predictions are often examined across different lead times. Short-term flood predictions typically include real-time, hourly, daily, and weekly forecasts. Real-time predictions cover periods ranging from a few minutes to an hour before a flood. Hourly predictions span 1-3 hours before a flood, and sometimes up to 18 or 24 hours. Daily predictions can forecast up to 1-6 days ahead of a flood, while monthly predictions can extend up to three months. Short-term predictions function as early warning systems, whereas long-term predictions are generally used for policy analysis. However, building an accurate lead time flood prediction model and identifying flood-prone areas is challenging due to frequent changes in climate patterns and data scarcity, which are common issues in hydrological research (Yu et al., 2018). Therefore, flood prediction often requires various data-specific simplified assumptions (Lohani et al., 2014).

Machine Learning (ML) and its subfield, Deep Learning (DL), have developed various advanced techniques for solving lead time prediction problems with better performance and cost-effectiveness, thereby overcoming the drawbacks of statistical methods for analyzing time series patterns (Mosavi et al., 2018; Maier et al., 2010; LeCun et al., 2015). DL has the ability to mimic the human brain and recognize patterns easily. Various DL methods, such as recurrent neural networks (RNN) (Pollack, 1990), long short term memory (LSTM) (Hochreiter & Schmidhuber, 1997), gated recurrent unit (GRU) (Cho et al., 2014), convolutional neural network (CNN) (Yann & Yoshua, 2002), have the capacity to extract past information from time series events, which can help in predicting future events. LSTM, in particular, is more powerful for analyzing time series or sequential problems due to its specialized gates. Studying these DL methods for implementing a flood forecasting model is necessary.

One of the key research problems in reducing flood damage is the creation of short-term forecasts or early warning systems, which is quite limited (Zhang et al., 2018; Mulia & Handayani, 2024). In general, it is claimed that using radar rainfall datasets to develop a prediction model yields superior accuracy (Maddox et al., 2002). To forecast daily runoff, Ghose (2018) develops a backpropagation neural network (BPNN) prediction model using daily water level data of the Brahmani River for two years (2013-2014) and observes good results. Based on monthly average rainfall time series data, Kumar et al. (2019) build a monthly precipitation forecasting model using an LSTM network that outperforms the RNN in their investigation and has the ability to evaluate climatic occurrences. Hamidreza et al. (2019) present an LSTM model for streamflow forecasting, utilizing historical meteorological and streamflow datasets, which yields better results than the physical model. In Ni et al. (2020), a hybrid model coupling LSTM with a convolutional neural network (CNN-LSTM) is used to construct a monthly streamflow and rainfall forecasting model, where the extraction of temporal features makes use of convolutional layers. When compared to MLP and plain LSTM, the results are shown to be superior. Using monthly evapotranspiration (ET), hourly rainfall, and hourly runoff data from two watersheds, Zhongrun et al. (2020) investigate the prediction performance of LSTM and sequence-to-sequence structures for hourly runoff. In terms of short-term flood prediction, this model performs better than traditional ML models. It is crucial to concentrate on the spatial and temporal data for studying ML/DL algorithms in the field of flood prediction and hydrological research rather than modifying the algorithm itself. Hence, the existing methods examined here use their specific datasets. Consequently, it is highly challenging to develop a benchmark model to predict floods everywhere in the world. As a result, the proposed flood prediction model is examined using a locally integrated dataset in this study.

The increasing frequency of floods in the Imphal Valley, Manipur, exacerbated by unusual rainfall patterns, urbanization, and changing land use, has exposed the inadequacy of existing flood forecasting systems. Current methods struggle with accuracy, particularly in regions with significant missing data and complex hydrological patterns. The lack of a reliable flood early warning system (FEWS) has led to devastating consequences, including loss of life, property damage, and agricultural disruption. This study addresses these challenges by proposing a hybrid approach by combining the KNN algorithm with LSTM which is capable of accurately forecasting river water levels in the Imphal Valley, thereby improving flood prediction and enabling timely early warnings.

This study aims to develop a robust data-driven approach for forecasting river water levels in the Imphal Valley, Manipur, using a hybrid KNN-LSTM framework. The approach addresses the challenges of missing data and leverages the strengths of KNN for data imputation and LSTM for analyzing complex spatial-temporal relationships in hydrological time series. The model's predictions are then integrated into an early warning algorithm, which assesses the risk of flooding and triggers alerts when water levels are forecasted to reach critical thresholds. This approach is particularly effective in scenarios with significant data gaps. Additionally, a thorough parameter analysis is conducted to evaluate how different inputs and structural configurations of the model influence the accuracy of forecasts across various lead times. By integrating KNN for data imputation and LSTM for time series analysis, this study presents a robust method for water level forecasting, particularly effective in scenarios where large portions of data are missing.

The rest of the paper is structured as follows: Section 2 describes the study area focused on in this study. Section 3 presents the methodology. Section 4 describes the dataset used in this study. Section 5 outlines the results and discussion. Finally, Section 6 provides the concluding remarks.

2. STUDY AREA

The Imphal Valley in Manipur, northeastern India, serves as the focal point of this study, as illustrated in Figure 1. The valley, situated roughly 790 meters above sea level and encompassing 1,843 square kilometres, serves as the state's socio-economic center encircled by hill ranges (Sophia & Devi, 2020). It is traversed by key rivers, including the Imphal, Iril, Nambul, and Thoubal, which are often overflowing during the monsoon season due to intense rainfall (Singh, 2022). The region experiences a subtropical climate, with an annual average rainfall of around 1,300 mm, primarily occurring during the monsoon season. These floods, triggered by rapid rises in river water levels, significantly disrupt agriculture, the primary occupation of the valley's inhabitants (Sophia & Devi, 2020).

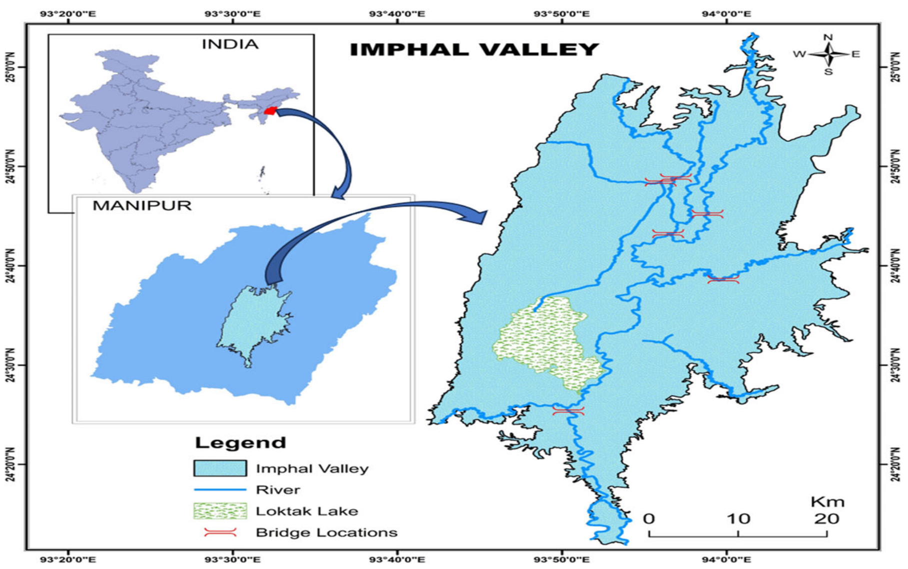


Figure 1. Location of Imphal Valley, highlighting bridge locations used as gauge stations for collecting river water level data

The Imphal Valley's susceptibility to flooding and its impact on the socio-economic conditions of the region's people motivate the development of a FEWS using advanced Machine Learning (ML) and Deep Learning (DL) methods, applied to the real-world scenario using the spatial dataset of this region. Over recent years, Manipur has witnessed significant climatic changes, particularly in rainfall patterns, which have profoundly impacted the region. To better understand these changes, rainfall data from nine major districts in Manipur are analyzed over a five-year period (2014-2018), based on data provided by the Directorate of Environment, Manipur (Figure 2). The analysis highlights that rainfall patterns have become more variable and high intensity during this period, which is likely to occur more unpredictable flood events in the region. These changes have resulted in several severe floods and landslides during this period, leading to substantial loss of life, infrastructure and property damage, agricultural land destruction, and significant environmental degradation. This study addresses the region's hydrological challenges, such as data scarcity and complex spatial and temporal dynamics, to contribute to more effective flood management in the valley.

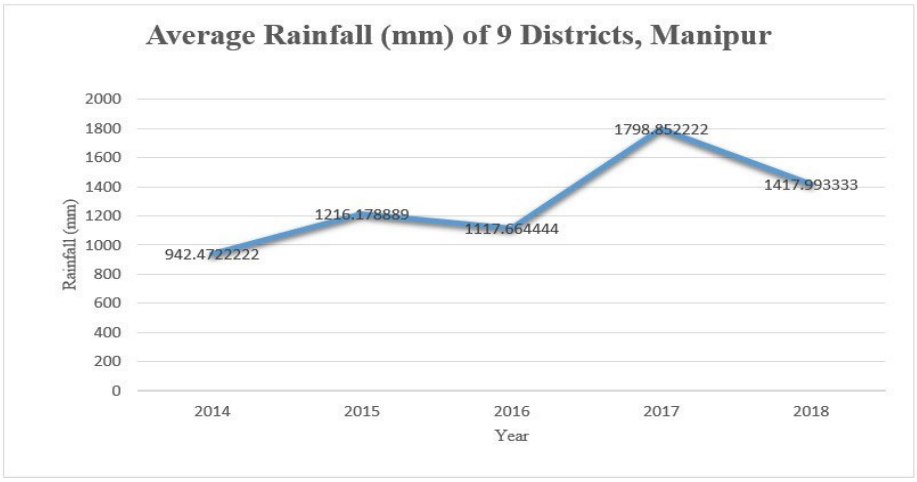


Figure 2. Rainfall patterns over the years (2014-2018)

3. METHODOLOGY

In this study, a hybrid model integrating KNN (Kulesh et al., 2008) with LSTM model is employed for predicting river water levels. The dataset used exhibits a large amount of missing data in the target variable, i.e., river water level, primarily occurring during the monsoon season. To address this issue, KNN is utilized to estimate the missing values based on the available data, using features such as rainfall, temperature, and relative humidity over a 5-day period. Once the missing values are imputed, the dataset is used to train an LSTM model. This imputation significantly improves model accuracy in predicting the river water level. The LSTM-based model is highly effective for predicting future river water levels, which is crucial for assessing the probability of future floods. The LSTM model is well-suited for handling sequential data and capturing temporal dependencies, essential for predicting future patterns in river water levels. This section explains the methodology used for implementing the proposed FEWS. First, the framework architecture of the proposed methods is presented, followed by a detailed explanation of the different components of the system architecture.

The proposed system architecture is structured into several key components, each contributing to the overall goal of flood forecasting and early warning. This section provides a detailed explanation of these components, outlining the processes involved in each stage. For a clearer understanding of how these elements interact, a conceptual framework is illustrated in Figure 3. This architecture comprises four modules: Data Collection, Data Manipulation, Model Implementation, and Early Warning and Flood Forecasting, each performing a crucial role in the system.

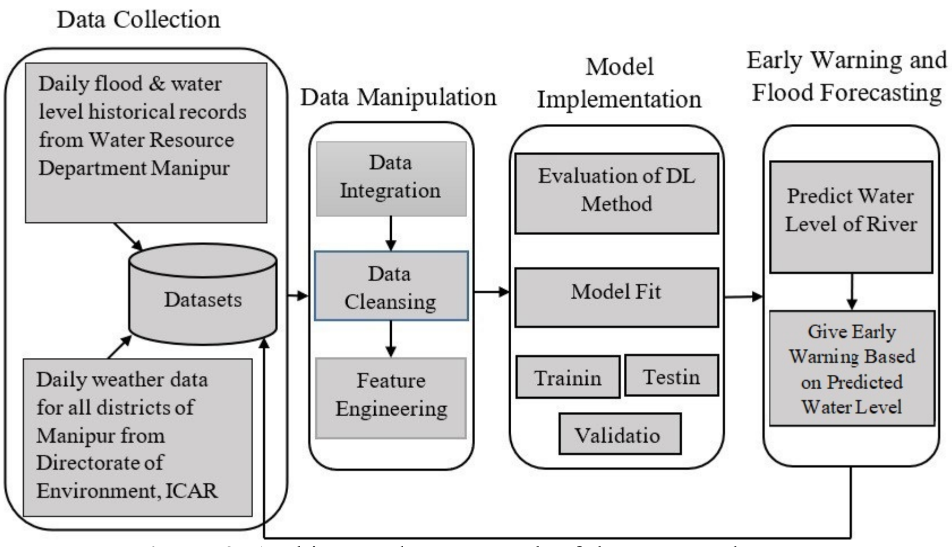

Data Collection
Daily flood & water level historical records from Water Resource Department Manipur
Datasets
Daily weather data for all districts of Manipur from Directorate of Environment, ICAR
Data Manipulation
Data Integration
Data Cleansing
Feature Engineering
Model Implementation
Evaluation of DL Method
Model Fit
Trainin
Testin
Validatio
Early Warning and Flood Forecasting
Predict Water Level of River
Give Early Warning Based on Predicted Water Level

Figure 3. Architectural Framework of the proposed system

3.1 Data Collection

The data process collection involves aggregating crucial flood-related information from various sources to ensure a comprehensive understanding of the factors contributing to flood events. This step is essential for gathering accurate data in a digital format, which is necessary for subsequent analysis and model development. In this study, daily district-level weather data provided by the Directorate of Environment, Manipur, along with water level measurements from six key gauge sites such as Minuthong Bridge, Lilongthong Bridge, Irilbung Bridge, Hump Bridge, Thoubal Bridge, and Ithai Barrage are utilized. These measurements, spanning from 2014 to 2018, monitor the main rivers within the valley region and are sourced from the Water Resource Department, Manipur. As the data sources come in diverse raw formats and structures, it is essential to subject the collected data to rigorous analysis, evaluation, and preprocessing. This approach ensures accurate labeling and standardization, allowing ML and DL algorithms to effectively process the data, thereby improving the prediction accuracy of the models.

3.2 Data Manipulation

The collected raw data undergoes a comprehensive data manipulation and preprocessing phase, essential for preparing the dataset for model development. This module is structured around three key processes: data integration, data cleansing, and feature engineering.

3.2.1 Data Integration

Data integration involves merging weather and hydrological data from various sources into a cohesive, unified dataset (Bergamaschi et al., 2011). This process is crucial for developing a comprehensive flood forecasting model, ensuring that all relevant information is combined effectively. In this study, data integration combines daily district-level weather data, including rainfall, temperature, and relative humidity, with hydrological data from six key gauge stations: Minuthong Bridge, Lilongthong Bridge, Irilbung Bridge, Hump Bridge, Thoubal Bridge, and Ithai Barrage. These stations monitor the main rivers in the Imphal Valley, and water level data from them spans from 2014 to 2018.

The integration process addresses challenges inherent in the collected datasets, which come from different sources such as the Directorate of Environment, Manipur, and the Water Resource Department, Manipur. These sources use various formats and measurement frequencies, such as daily weather records versus different intervals for water level measurements. To ensure consistency, the process standardizes measurements into uniform units and temporal resolutions. Synchronizing the datasets based on a common temporal reference aligns weather and water level data, allowing the model to effectively learn relationships between precipitation events and water level changes. Additionally, the integration phase identifies and eliminates redundancies, such as duplicate "datetime" or location identifiers, thereby streamlining the dataset and improving the efficiency and accuracy of subsequent analyses.

3.2.2 Data Cleansing and Feature Engineering

After integrating the data from multiple sources, a comprehensive data cleansing process is undertaken to enhance the quality of the dataset by identifying and correcting errors and inconsistencies (Borrohou et al., 2023). This includes identifying and correcting errors, standardizing formats, and ensuring consistent measurement units. A key aspect of data cleansing in this study is outlier filtering. This method involves calculating the mean and standard deviation of the river water level data. Any data points falling outside three standard deviations from the mean are considered outliers and are replaced with threshold values (either the upper or lower limit) to ensure that extreme values do not adversely affect model training. By filtering out these outliers, the dataset becomes more robust, improving the accuracy and reliability of the predictive models.

Feature engineering further refines the dataset by selecting and creating relevant features to improve model performance. In this study, features were chosen based on their strong correlation with river water levels, ensuring that the most informative inputs are used.

Additionally, a dummy function similar to one-hot encoding was employed to transform categorical station identifiers into binary features, allowing the model to account for spatial variations in river water levels. This approach enhances the model's ability to accurately predict river water levels and supports more reliable flood forecasting in the Imphal Valley.

3.3 Model Implementation

In this section, various ML/DL algorithms are implemented and evaluated for predicting river water levels, utilizing a meticulously preprocessed dataset. Significant data challenges are addressed, including the imputation of missing water level values using the KNN algorithm, which is crucial for maintaining the integrity of the input data. Following these preparations, the LSTM, ANN, and 1D CNN models are applied to the dataset. Each model is rigorously evaluated to determine its effectiveness in capturing the temporal dynamics of input features, such as rainfall, temperature, and relative humidity, to predict river water levels across multiple gauge stations within the Imphal Valley.

3.3.1 K-Nearest Neighbors (KNN) Algorithm for Handling Missing Value

A specific challenge in this study is handling the water level data provided by the Water Resource Department, which contained significant missing values, as water levels are only recorded during the monsoon seasons. Since these missing values occur exclusively in the target variable (river water level), traditional non-ML imputation methods are inadequate for handling such a large proportion of missing data (Weerakody et al., 2021). To address this challenge, the KNN algorithm (Kulesh et al., 2008) is employed to impute the missing values. The KNN model is trained on a carefully selected subset of the dataset that contains no missing data in the target variable, ensuring that the model is trained on complete and reliable data. This subset utilized a five-day lag of key meteorological features, including rainfall, temperature, and relative humidity.

The rationale behind using a five-day lag is to capture the temporal dynamics between weather conditions and river water levels, as the impact of rainfall and temperature on water levels typically unfolds over several days rather than immediately. Training the KNN model on this lagged data could effectively learn the relationships between these features and the water level, enabling accurate imputation of the missing values. The KNN algorithm operates by finding the nearest data points to a given instance based on a distance metric, which in this case was the Euclidean distance. The Euclidean distance is calculated as the straight-line distance between two points in multi-dimensional space, defined mathematically as:

$$Distance(x, y) = \sqrt{\sum_{i=0}^{n} (y_i - x_i)^2} \tag{1}$$

Here, x and y are the two points, i ∈ n number of coordinates, and x_i and y_i represent the coordinates of these points. By applying this method, the model is able to effectively estimate the missing water level values, thereby enhancing the completeness and reliability of the dataset for subsequent predictive modeling.

3.3.2 Long Short Trem Memory (LSTM)

LSTM model (Hochreiter & Schmidhuber, 1997) is a specialized form of RNN designed to address the limitations of traditional RNNs, particularly the challenge of long-term temporal dependencies and the vanishing gradient problem (Houdt et al., 2020). LSTM networks are capable of retaining important temporal information from the input data over extended sequences, making them particularly suitable for time series forecasting tasks like predicting water levels.

In this method, a multivariate, multiple-time-series LSTM model is employed, incorporating lagged data as input features. Specifically, we use a lag of 3, 5, 7, and 10 days for the features: rainfall, temperature, relative humidity, and water level from six gauge stations. The goal is to predict the next day's water level at each station. These features are selected due to their strong correlation with flood events in Imphal valley, Manipur, which are predominantly triggered by heavy precipitation. Temperature and relative humidity are also closely associated with precipitation patterns, further justifying their inclusion in the model. Each time series in the dataset spans five years (2014-2018) and consists of daily records. For each record, we aimed to predict the water level at time t+1 using data from t and earlier. To accommodate the temporal dependencies, the data is reframed so that the input to the model for each time step consisted of a matrix of size 5×N, where N represents the number of features. This process is applied across all six gauge stations, ensuring that the model is fed with a consistent and structured dataset.

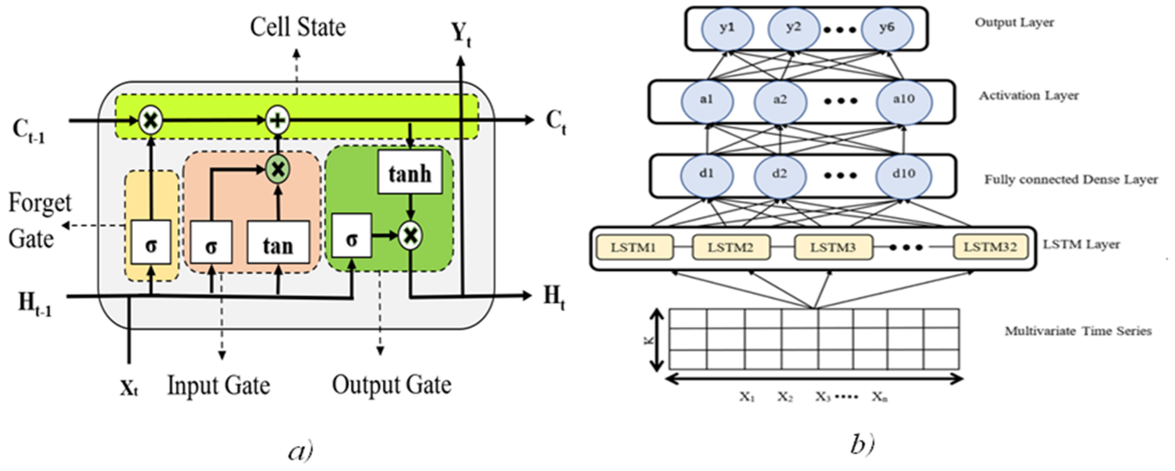


Figure 4. (a) Internal structure of an LSTM unit (b) Overall architecture of the LSTM model

The internal structure of an LSTM unit and the overall architecture of the proposed LSTM model are illustrated in Figure 4 above. The model begins with an input layer that accepts data with a shape of 30×10, corresponding to the features and the lagged time steps. This is followed by an LSTM layer comprising 32 neurons, which is responsible for processing the sequential data. The LSTM layer is connected to a fully connected dense layer with an activation function, which helps in transforming the output of the LSTM layer into a suitable format for the final prediction. The output layer consists of six neuron, representing the predicted water level for the next day at each station. We employ the Adam optimizer (Kingma & Ba, 2014) to train the model, which is well-regarded for its efficiency in optimizing neural networks by adjusting the learning rate adaptively throughout the training process.

3.3.3 Artificial Neural Network (ANN) with Backpropagation Method

The Artificial Neural Network (ANN) (Kim et al., 2016) model is a core architecture in deep learning, designed to emulate the way the human brain processes information. In this study, the ANN is employed as one of the baseline models for predicting river water levels across multiple stations. The architecture (Figure 5) comprises an input layer, hidden layers, and an output layer, with each neuron in one layer fully connected to neurons in the subsequent layer. The ANN model is configured to capture the complex relationships between input features such as rainfall, temperature, and relative humidity across a five-day lag, and the target variable, which is the river water level at six stations. The input features are processed through hidden layers, where each neuron applies a linear transformation followed by a ReLU (Rectified Linear Unit) activation function to extract non-linear patterns from the data. The output layer is configured to provide simultaneous predictions for river water levels at all six stations. The model is trained using the backpropagation algorithm to minimize the Mean Squared Error (MSE) loss function. Backpropagation (Rumelhart, 1987) ensures that the model learns effectively by adjusting the weights during training to reduce prediction errors.

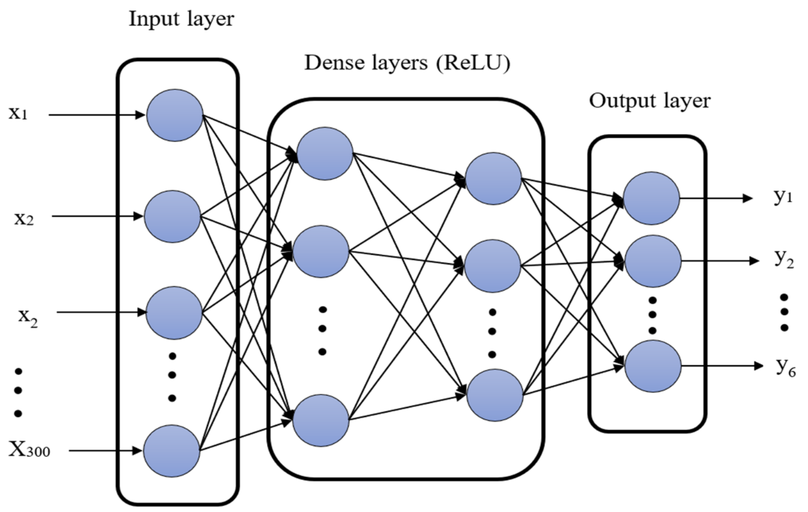


Figure 5. Architectural diagram of ANN

3.3.4 One-Dimensional Convolutional Neural Network (1D CNN)

The 1D CNN (Yann & Yoshua, 2002) model is an effective deep learning architecture designed for analyzing sequential data, such as time series from meteorological measurements. In this study, the 1D CNN is utilized to predict river water levels at multiple stations by processing sequences of input data that include features like rainfall, temperature, and relative humidity. The model architecture (Figure 6) starts with a convolutional layer that scans the input sequence to identify key patterns and relationships within the data. This layer is followed by a MaxPooling operation, which reduces the dimensionality of the data by focusing on the most significant features and helps in making the model more robust and efficient. Subsequent to the pooling layer, the processed data is flattened and passed through a fully connected layer. This layer integrates the features extracted by the convolutional and pooling layers to prepare them for the final prediction step. The output layer of the model provides simultaneous predictions for river water levels across the different stations. The model is trained using the Adam optimizer with the Mean Squared Error (MSE) as the loss function. This approach aims to minimize prediction errors and improve the accuracy of river level forecasts.

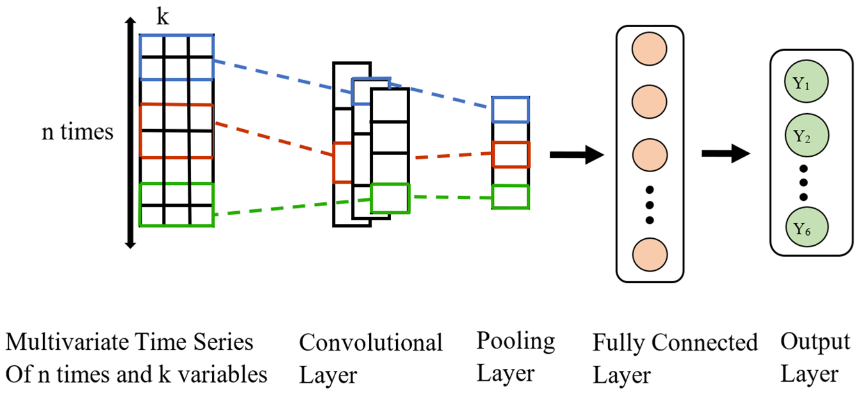


Figure 6. Architectural diagram of 1D CNN

3.3.5 Evaluation Metrics

The model's performances are thoroughly assessed using three essential evaluation metrics: Nash-Sutcliffe Efficiency (NSE), Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE). These metrics, detailed in Table 1, provide an intricate evaluation of the model's predictive accuracy and reliability. NSE measures the goodness of fit between observed and predicted river water levels, providing insight into the model's overall efficiency. MAPE quantifies the model's prediction accuracy by expressing the error as a percentage, making it easier to interpret the magnitude of the errors relative to the observed data. RMSE, on the other hand, highlights the model's ability to predict values close to the observed ones by penalizing larger errors more heavily. Together, these metrics deliver a comprehensive evaluation of the model's capability to accurately forecast river water levels, ensuring that it meets the necessary standards for reliable flood prediction.

Table 1. Definition of metrics used for network's performances

Metrics	Formulae	Description
NSE	$1 - \left\{\frac{\sum_{t=1}^{T}(Q_t - O_t)^2}{\sum_{t=1}^{T}(Q_t - \bar{Q})^2}\right\}$	Here, Q_t and O_t are the observed and simulated data at the time step t and Q^{-} is the mean value of the observed data.
MAPE	$\frac{1}{T}\sum_{t=1}^{T}\left(\left\lvert\frac{y_t - p_t}{y_t}\right\rvert\right)100\%$	here, y_t and p_t are the observed and simulated data at the time step t.
RMSE	$\sqrt{\frac{\sum_{t=1}^{T}(y_t - p_t)^2}{T}}$	

3.4 Early Warning and Flood Forecasting

The final stage of the proposed framework involves the implementation of an algorithm designed to provide early warnings for potential floods. This algorithm leverages the predictive capabilities of the LSTM model developed in this study to forecast river water levels and issue alerts based on those predictions. The primary objective of this algorithm is to deliver timely alerts to communities at risk of flooding, thereby mitigating the potential damage caused by rising water levels. The algorithm operates by first assessing the continuity of rainfall, a critical factor in flood risk assessment. It then uses observed data comprising rainfall, temperature, relative humidity, and the current water level as inputs to the LSTM model. This model predicts the river water level for the next time step for all six gauge station simultaneously. One-step-ahead prediction of multiple stations is a complex task, as it requires capturing both the temporal dependencies of each station and the spatial correlations between them. The LSTM model is particularly well-suited for this challenge, as it efficiently handles multivariate time series data, learning temporal patterns and interdependencies across stations. Once the prediction is made, the algorithm compares the predicted water level with predefined thresholds for warning, flood, and high flood levels, as established by the Water Resource Department, Manipur. Based on this comparison, the algorithm generates an output that indicates the flood risk level. This output can then be used to issue warnings to the public, enabling proactive measures to prevent or minimize flood damage. The algorithm is designed as a generalized framework for a single station, making it adaptable and applicable to all six stations included in the study. This approach remains consistent and scalable across multiple gauge stations, enhancing its utility for comprehensive flood risk management.

The algorithm given here has an approximate complexity of O(n), where n is the length of the rainfall data (R). This is due to the fact that step 1 is a loop that continues as long as R is available, with step 2 involving the prediction of river water level using a proposed LSTM network, which typically has a time complexity of O(n), where n denotes the length of the input sequence, and steps 3 to 8 comparing the predicted river water level with predefined warning levels, flood levels, and high flood levels, with complexity of O(1), and step9 is an else statement with an O(1) complexity that is executed when steps 3 to 8 are not satisfied. Step 12 performs a simple comparison to check whether rainfall continues or not, which has a constant time complexity of O(1). Step 13 involves monitoring the current water level, which also has a constant time complexity of O(1). Step 14 checks whether the current water level is rising as compared to the previous water level, maintaining an O(1) complexity. If the water level is rising, step 15 invokes the prediction of the next day's water level using the proposed LSTM network, which has a time complexity of O(n). Subsequently, the comparison process (steps 3 to 10) is repeated, where each step has an O(1) complexity. If the current water level is not rising, step 17 returns a normal water level alert, which also has an O(1).

Algorithm 1. Flood early warning and forecasting

```
Input: Rainfall data (R), Water Level data (Warning Level, Flood Level, and High Flood Level)
Output: Alert current and future water level and flood situation to the Authorities

Step 1.  for R continue do
Step 2.      Predict (Water Level) using the proposed LSTM network;
Step 3.      if Water Level ≥ Warning Level and Water Level < Flood Level then
Step 4.          return (alert Warning level);
Step 5.      else if Water Level ≥ Flood Level and Water Level< High Flood Level then
Step 6.          return (alert Flood level);
Step 7.      else if Water Level ≥ High Flood Level then
Step 8.          return (alert High Flood level);
Step 9.      else
Step 10.         return (normal water level);
Step 11. end for
Step 12. if R==0 (Rainfall Stop) then
Step 13.     Monitor (Current Water Level)
Step 14.     if Current Water Level > Previous Water Level (Rising water level) then
Step 15.         Proceed the prediction of next day Water Level using proposed LSTM network and
                 perform the comparison process through steps 3 to 10
Step 16.     else
Step 17.         return (normal water level);
```

4. DATASET

This study utilizes a dataset containing meteorological and hydrological data from six gauge stations in the Imphal Valley, Manipur, India. The dataset spans five years (2014-2018) and includes daily measurements of rainfall, temperature, relative humidity, and river water level. The river water level serves as the target variable for prediction, while the other variables are used as inputs. The multi-station dataset provides a comprehensive representation of the hydrological and meteorological conditions in the region, making it suitable for developing and validating flood forecasting models. The river water level data contains approximately 52% missing values due to the limitation of observation, particularly during the non-monsoon months. To fill these gaps, the K-Nearest Neighbor (KNN) algorithm is employed to impute missing values based on five days of rainfall, temperature, and relative humidity, which are complete and free from missing values.

To capture temporal dependencies, the dataset is transformed into lagged input sequences of 3, 5, 7, and 10 days. For example, a 5-day lag incorporates daily values of rainfall, temperature, relative humidity, and river water levels from the previous five days. The input size for the model varies based on the selected lag. This transformation ensures that the model utilizes sufficient historical information to learn temporal and spatial patterns effectively.

The dataset is divided into three subsets for training, validation, and testing. Specifically, the first 70% of the data from the start of each gauge station's dataset is used for training the model. The subsequent 15%, covering the range from 70% to 85%, is set aside for validation during training. The remaining 15%, from 85% to 100%, is treated as unseen test data and is used for evaluating the model’s performances. This sequential slicing ensures that the models are trained, validated and evaluated by preserving the temporal ordering of the time-series dataset.

To capture the spatial uniqueness of each station, a dummy variable-based feature encoding is utilized, where each station is represented by a binary bit (Garavaglia et al., 1998). This encoding allows the model to differentiate data from the stations while preserving their spatial characteristics. By integrating these dummy-encoded features with lagged measurements of meteorological and hydrological variables, the dataset enables the model to effectively learn both local station-specific dynamics and broader regional trends, enhancing its predictive capabilities.

5. RESULTS AND DISCUSSION

An extensive comparative analysis is conducted by experimenting with various configurations of the LSTM network on the same dataset. The aim is to determine the most effective parameter settings that maximize predictive performance. The parameters vary, including input size, the number of hidden layers and neurons, activation functions, and the structure of the output layer.

The input size for the LSTM model is determined by two factors: the number of features and the length of the time lag sequence fed into the model. For each gauge station, we consider time lags of 3, 5, 7, and 10 days. Consequently, the input sizes configured for training the LSTM network are calculated as three days lag: 3×6=18, five days lag: 5×6=30, seven days lag: 7×6=42, and ten days lag: 10×6=60. These values represent the time sequence length multiplied by the number of gauge stations, resulting in a total input size of 18, 30, 42, and 60 sequences, respectively. The dataset itself contains 10 features, including the gauge sites, which are used as inputs to the model.

The LSTM, ANN and 1D CNN model are configured with varying complexities to test their performance. We experiment with networks containing a single hidden layer, with the number of neurons set to 16, 32, and 64. These configurations are chosen to explore the impact of network depth and neuron count on the model’s ability to capture temporal dependencies in the data. The performance of these configurations of each model is evaluated using key metrics such as NSE, MAPE, and RMSE, as defined in Table 2, 3, and 4. Among the different configurations, the LSTM model with a 5-day time lag and a single hidden layer of 32 neurons demonstrated the lowest error and highest efficiency, indicating that this setup is the most effective for predicting river water levels. In contrast, the ANN with 64 neurons and 1D CNN with 16 filters achieve their best performance using a 3-day lag. This indicates that both the ANN and CNN are effective for next step prediction with shorter time sequences. Figure 7, 8, and 9 presents a comparison between the actual and predicted river water levels of the training, validation, test data, generated by different configurations of the LSTM, ANN, 1D CNN model respectively. Since the models predict the river water levels for all the six tations simultaneously, the graphs are based on the averaged water levels across all the stations for each time step. This comparison highlights the effectiveness of each model’s optimal configuration in forecasting river water levels, allowing for a detailed assessment of their predictive accuracy against the observed target values.

For further validation, the performance of the proposed model is compared with two conventional models: ANN and 1D CNN, as discussed in this study. The results indicate that the proposed KNN-LSTM model outperforms both the 1D CNN and ANN models in predicting river water levels across all configurations. Specifically, for the 5-day lag with 32 neurons, the KNN-LSTM model achieved a testing NSE of 0.9393, a MAPE of 0.1076, and an RMSE of 1.1828, demonstrating superior accuracy across all metrics. In comparison, the testing performance of the ANN model resulted in an NSE of 0.8765, a MAPE of 0.1528, and an RMSE of 1.6237, while the 1D CNN model obtained an NSE of 0.8864, a MAPE of 0.1456, and an RMSE of 1.6186. Table 5 presents the comparison of training, validation, and testing performances for the optimized configuration of the proposed and conventional models. These results highlight the robustness and effectiveness of the KNN-LSTM approach in handling the complex temporal and spatial dynamics present in the dataset. Figure 10 illustrates the comparison between the actual and predicted river water levels for training, validation, and testing data of each model discussed in Table 5, clearly demonstrating the superior predictive accuracy of the KNN-LSTM model. The graphical representation further reinforces its potential as a reliable tool for flood forecasting in the context of this study.

Table 2. Comparison of the configured parameter's performances of LSTM network

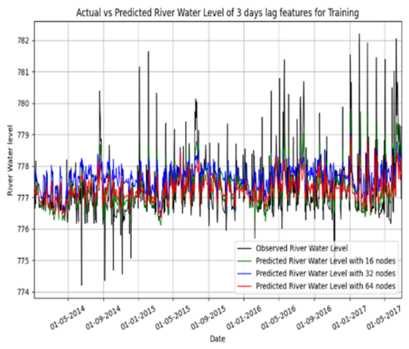
Model	Day Lag	Configuration	Dataset Split	NSE	MAPE	RMSE
LSTM	3-day lag	16 Neurons	Training	0.9293	0.1035	1.2507
			Validation	0.9216	0.1219	1.3581
			Testing	0.9123	0.1351	1.4221
		32 Neurons	Training	0.9174	0.1231	1.3520
			Validation	0.9172	0.1318	1.3964
			Testing	0.9008	0.1418	1.5128
		64 Neurons	Training	0.9265	0.1051	1.2754
			Validation	0.9180	0.1224	1.3896
			Testing	0.8907	0.1503	1.5876
	5-day lag	16 Neurons	Training	0.9258	0.1171	1.3023
			Validation	0.9233	0.1259	1.3201
			Testing	0.9161	0.1315	1.3914
		32 Neurons	Training	**0.9454**	**0.0869**	**1.0988**
			Validation	**0.9445**	**0.0941**	**1.1421**
			Testing	**0.9393**	**0.1076**	**1.1828**
		64 Neurons	Training	0.9251	0.1023	1.2881
			Validation	0.9133	0.1222	1.4277
			Testing	0.8893	0.1511	1.5975
	7-day lag	16 Neurons	Training	0.9249	0.1059	1.2891
			Validation	0.9169	0.1225	1.3981
			Testing	0.9161	0.1327	1.3899
		32 Neurons	Training	0.9336	0.0997	1.2123
			Validation	0.9284	0.1177	1.2969
			Testing	0.9171	0.1302	1.3817
		64 Neurons	Training	0.9234	0.1073	1.3019
			Validation	0.9162	0.1226	1.4034
			Testing	0.9098	0.1311	1.4415
	10-day lag	16 Neurons	Training	0.9311	0.1006	1.2358
			Validation	0.9265	0.1185	1.3125
			Testing	0.9133	0.1336	1.4146
		32 Neurons	Training	0.9310	0.1011	1.2351
			Validation	0.9261	0.1188	1.3167
			Testing	0.9175	0.1338	1.3802
		64 Neurons	Training	0.9286	0.1108	1.2567
			Validation	0.9251	0.1276	1.3251
			Testing	0.9209	0.1219	1.3508

Table 3. Comparison of the configured parameter's performances of ANN network

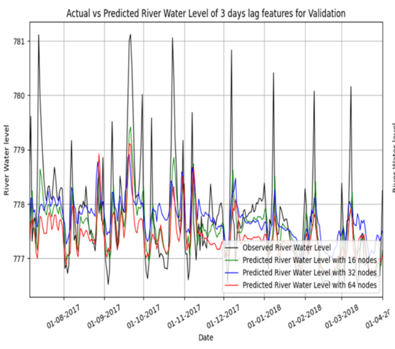
Model	Day Lag	Configuration	Dataset Split	NSE	MAPE	RMSE
ANN	3-day lag	16 Neurons	Training	0.8951	0.1416	1.5232
			Validation	0.8947	0.1444	1.5592
			Testing	0.8863	0.1546	1.6199
		32 Neurons	Training	0.8975	0.1383	1.5057
			Validation	0.8971	0.1453	1.5564
			Testing	0.8885	0.1543	1.6041
		64 Neurons	Training	**0.8990**	**0.1367**	**1.4951**
			Validation	**0.8979**	**0.1421**	**1.5506**
			Testing	**0.8944**	**0.1449**	**1.5608**
	5-day lag	16 Neurons	Training	0.8805	0.1524	1.6264
			Validation	0.8789	0.1581	1.6871
			Testing	0.8723	0.1652	1.7159
		32 Neurons	Training	0.8851	0.1501	1.6621
			Validation	0.8849	0.1553	1.6451
			Testing	0.8765	0.1528	1.6237
		64 Neurons	Training	0.8925	0.1425	1.5420
			Validation	0.8892	0.1431	1.5466
			Testing	0.8864	0.1533	1.6183
	7-day lag	16 Neurons	Training	0.8970	0.1396	1.5097
			Validation	0.8960	0.1469	1.5638
			Testing	0.8829	0.1572	1.6426
		32 Neurons	Training	0.8992	0.1385	1.4937
			Validation	0.8935	0.1481	1.5822
			Testing	0.8872	0.1529	1.6123
		64 Neurons	Training	0.8871	0.1470	1.5805
			Validation	0.8858	0.1535	1.6385
			Testing	0.8723	0.1637	1.7155
	10-day lag	16 Neurons	Training	0.8789	0.1548	1.6498
			Validation	0.8770	0.1615	1.6854
			Testing	0.8674	0.1683	1.7499
		32 Neurons	Training	0.8981	0.1395	1.5153
			Validation	0.8962	0.1460	1.5459
			Testing	0.8822	0.1560	1.6491
		64 Neurons	Training	0.8978	0.1403	1.5086
			Validation	0.8971	0.1465	1.5450
			Testing	0.8790	0.1602	1.6713

Table 4. Comparison of the configured parameter's performances of 1D CNN network

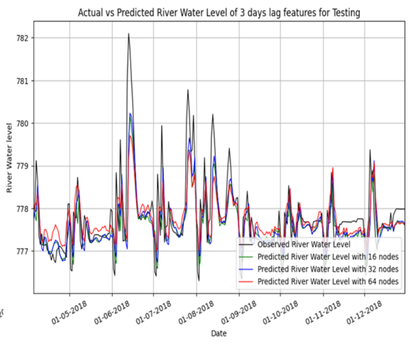
Model	Day Lag	Configuration	Dataset Split	NSE	MAPE	RMSE
1D-CNN	3-day lag	16 Filters	Training	**0.9276**	**0.1054**	**1.2911**
			Validation	**0.9167**	**0.1239**	**1.3999**
			Testing	**0.9111**	**0.1327**	**1.4320**
		32 Filters	Training	0.9271	0.1060	1.2701
			Validation	0.9164	0.1262	1.4024
			Testing	0.9085	0.1384	1.4529
		64 Filters	Training	0.8810	0.1496	1.6225
			Validation	0.8697	0.1641	1.7514
			Testing	0.8669	0.1667	1.7522
	5-day lag	16 Filters	Training	0.9224	0.1108	1.3099
			Validation	0.9109	0.1302	1.4468
			Testing	0.8973	0.1426	1.5389
		32 Filters	Training	0.9154	0.1133	1.3677
			Validation	0.8971	0.1336	1.5554
			Testing	0.8864	0.1456	1.6186
		64 Filters	Training	0.9128	0.1214	1.3888
			Validation	0.9025	0.1372	1.5138
			Testing	0.8924	0.1443	1.5749
	7-day lag	16 Filters	Training	0.9121	0.1189	1.3943
			Validation	0.8949	0.1386	1.5720
			Testing	0.8867	0.1443	1.6157
		32 Filters	Training	0.9067	0.1307	1.4368
			Validation	0.8966	0.1483	1.5593
			Testing	0.8804	0.1615	1.6602
		64 Filters	Training	0.9045	0.1332	1.4538
			Validation	0.8949	0.1508	1.5718
			Testing	0.8842	0.1602	1.6334
	10-day lag	16 Filters	Training	0.9131	0.1216	1.3863
			Validation	0.9022	0.1396	1.5143
			Testing	0.8860	0.1507	1.6225
		32 Filters	Training	0.9114	0.1260	1.4001
			Validation	0.9038	0.1429	1.5021
			Testing	0.8952	0.1500	1.5557
		64 Filters	Training	0.9119	0.1252	1.3961
			Validation	0.9027	0.1403	1.5105
			Testing	0.8865	0.1523	1.6188



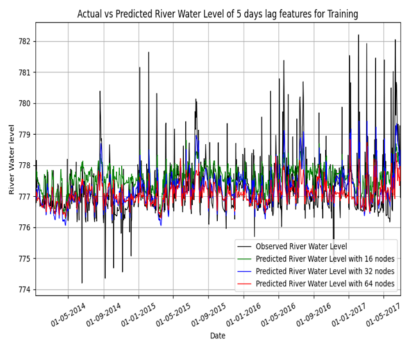
(a)Training graph of 3-day lag



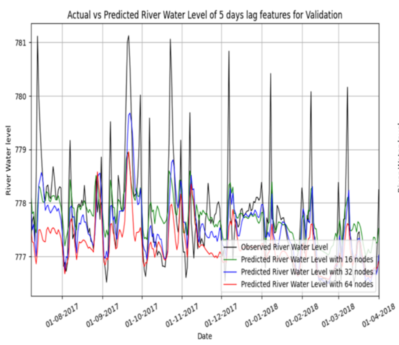
(e)Validation graph of 3-day lag



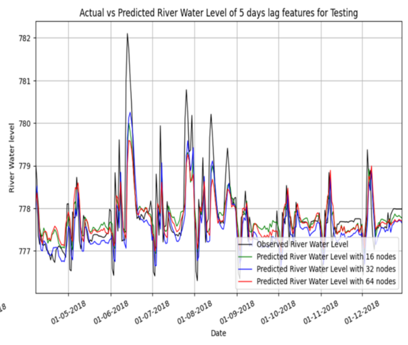
(i)Testing graph of 3-day lag



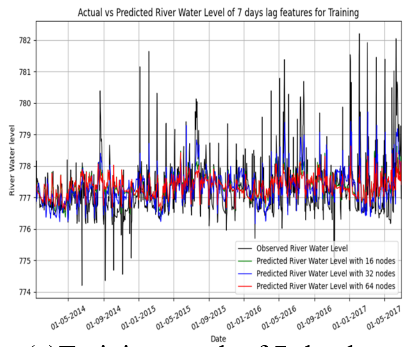
(b)Training graph of 5-day lag



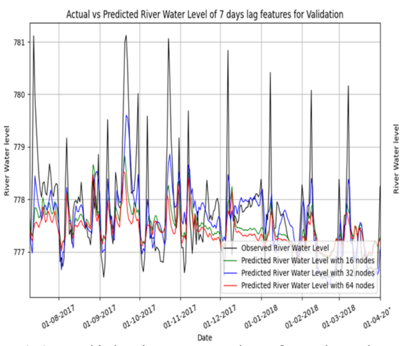
(f)Validation graph of 5-day lag



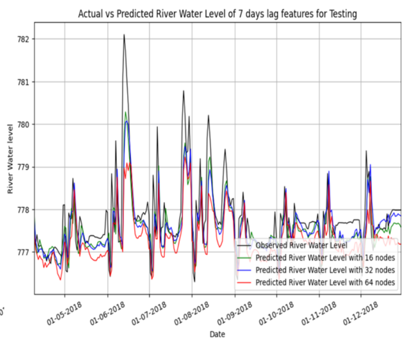
(j)Testing graph of 5-day lag



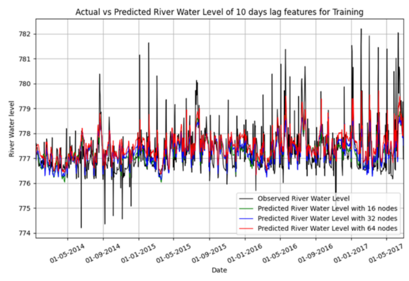
(c)Training graph of 7-day lag



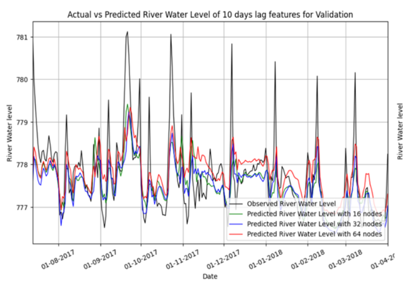
(g)Validation graph of 7-day lag



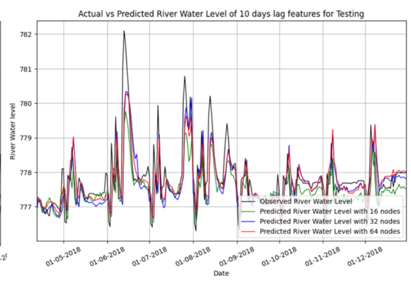
(k)Testing graph of 7-day lag



(d)Training graph of 10-day lag

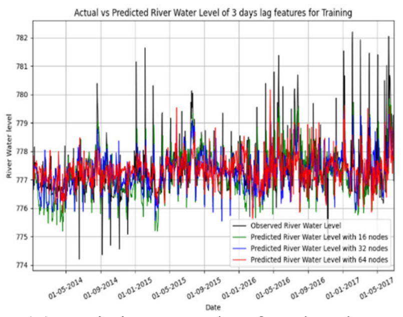


(h)Validation graph of 8-day lag

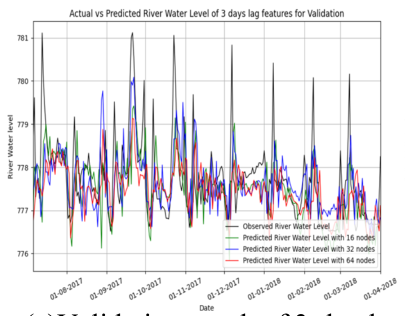


(l)Testing graph of 10-day lag

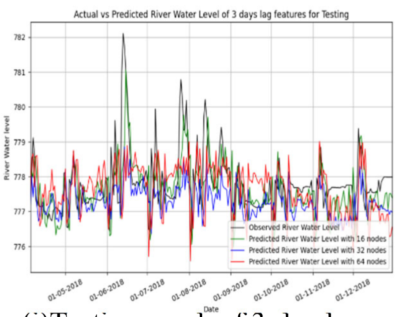
Figure 7. Figures (a) to (l) illustrate the actual and predicted river water levels for different lag periods of 3, 5, 7, and 10 days, using a multivariate multiple-time-series LSTM model with 16, 32, and 64 nodes.



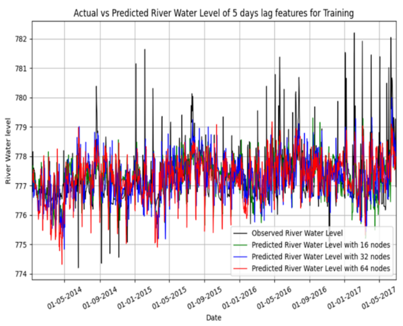
(a)Training graph of 3-day lag



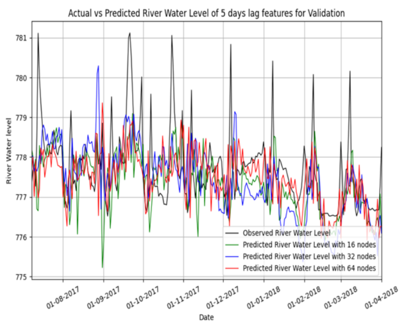
(e)Validation graph of 3-day lag



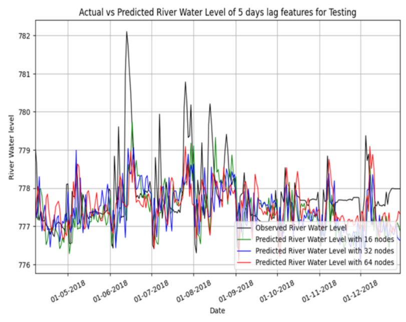
(i)Testing graph of 3-day lag



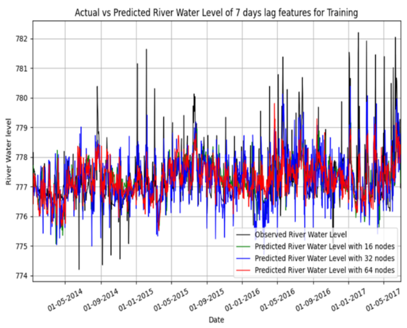
(b)Training graph of 5-day lag



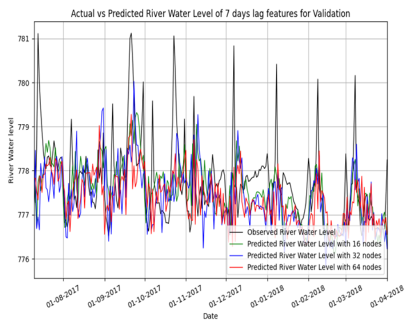
(f)Validation graph of 5-day lag



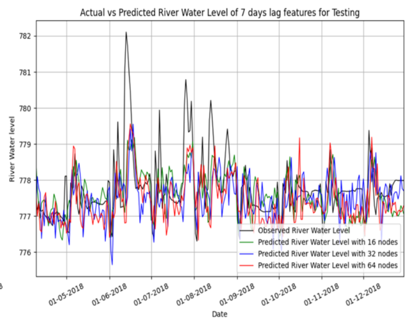
(j)Testing graph of 5-day lag



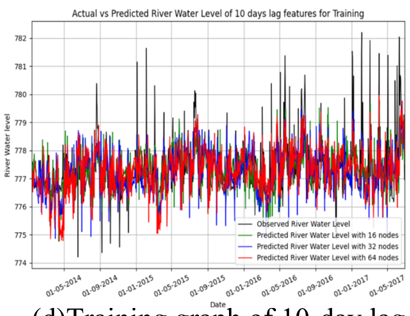
(c)Training graph of 7-day lag



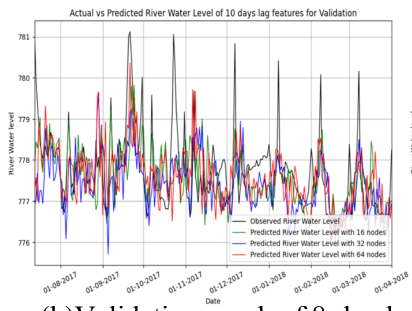
(g)Validation graph of 7-day lag



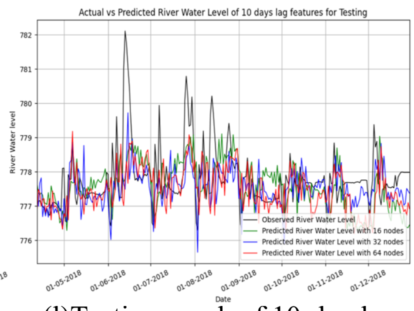
(k)Testing graph of 7-day lag



(d)Training graph of 10-day lag

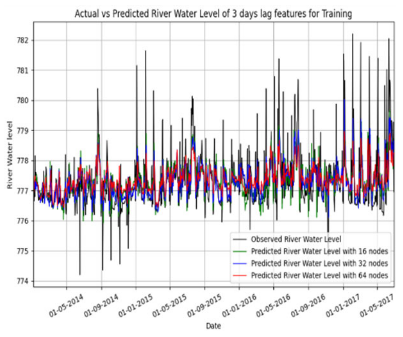


(h)Validation graph of 8-day lag

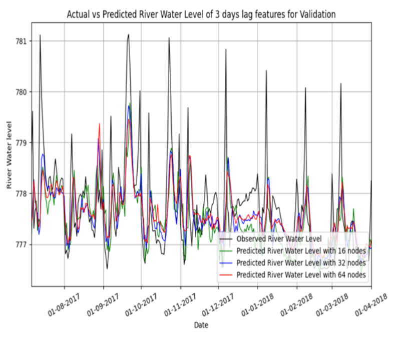


(l)Testing graph of 10-day lag

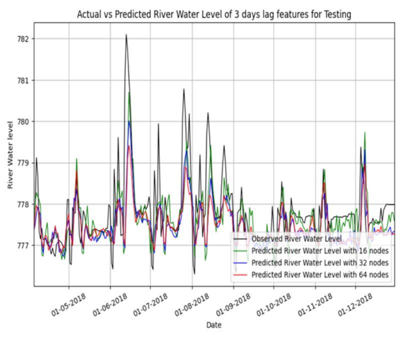
Figure 8. Figures (a) to (l) illustrate the actual and predicted river water levels for different lag periods of 3, 5, 7, and 10 days, using ANN model with 16, 32, and 64 nodes.



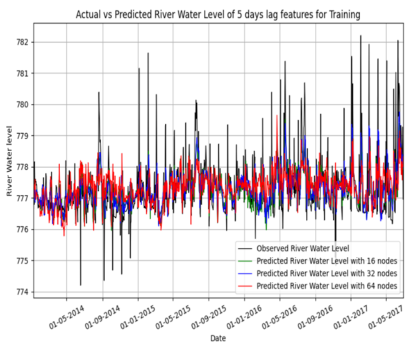
(a)Training graph of 3-day lag



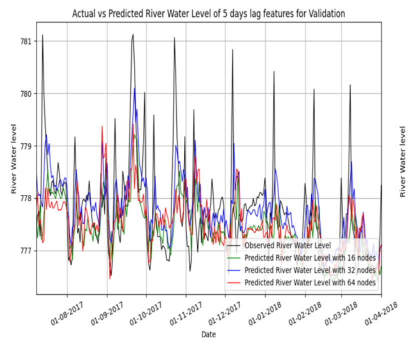
(e)Validation graph of 3-day lag



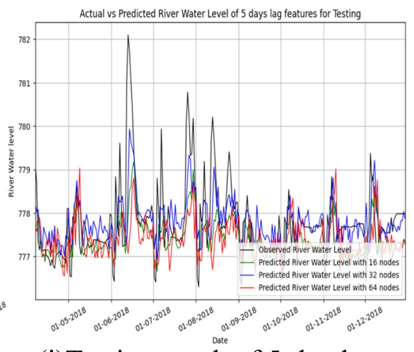
(i)Testing graph of 3-day lag



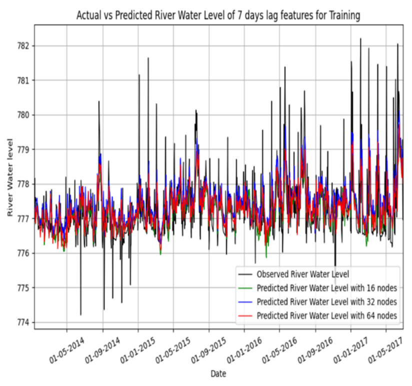
(b)Training graph of 5-day lag



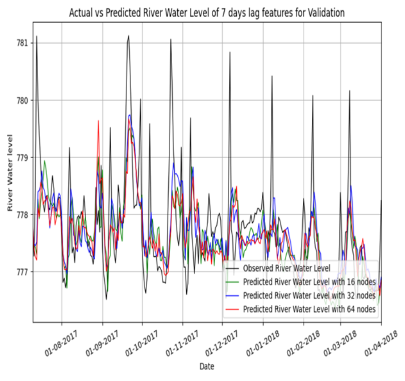
(f)Validation graph of 5-day lag



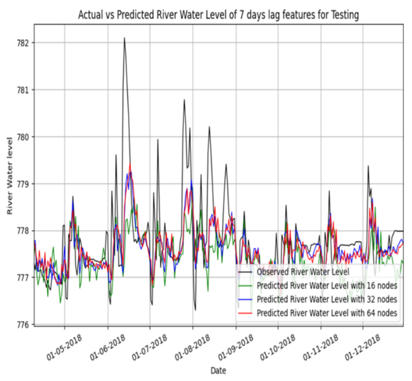
(j)Testing graph of 5-day lag



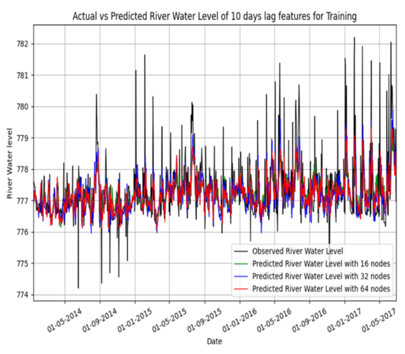
(c)Training graph of 7-day lag



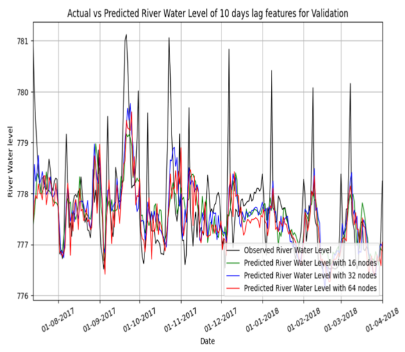
(g)Validation graph of 7-day lag



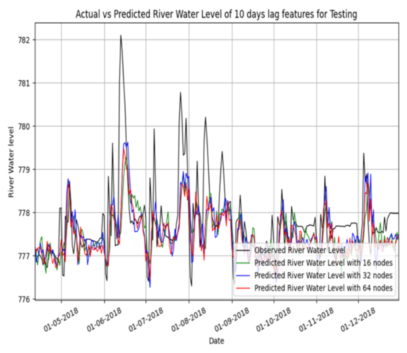
(k)Testing graph of 7-day lag



(d)Training graph of 10-day lag

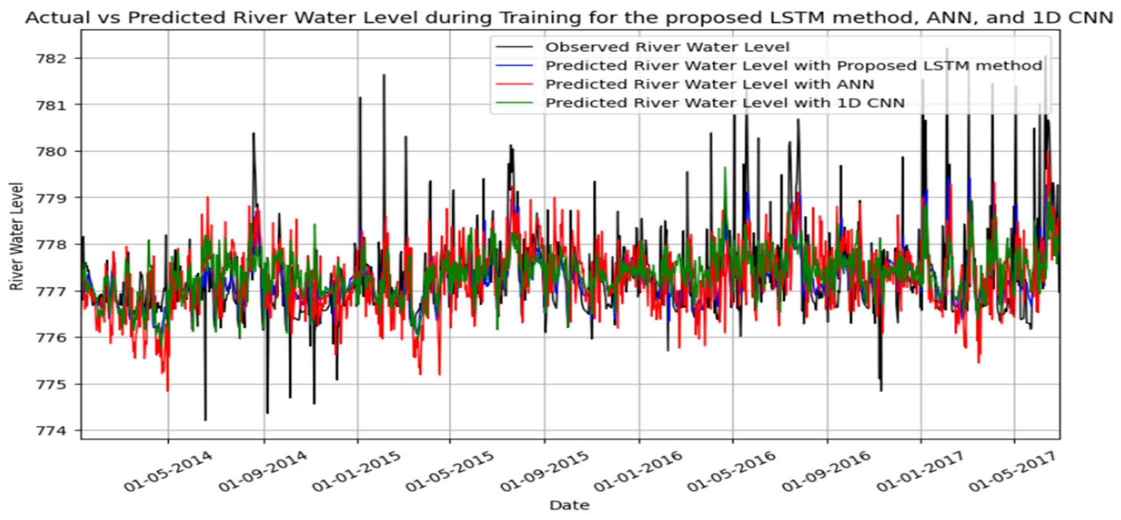


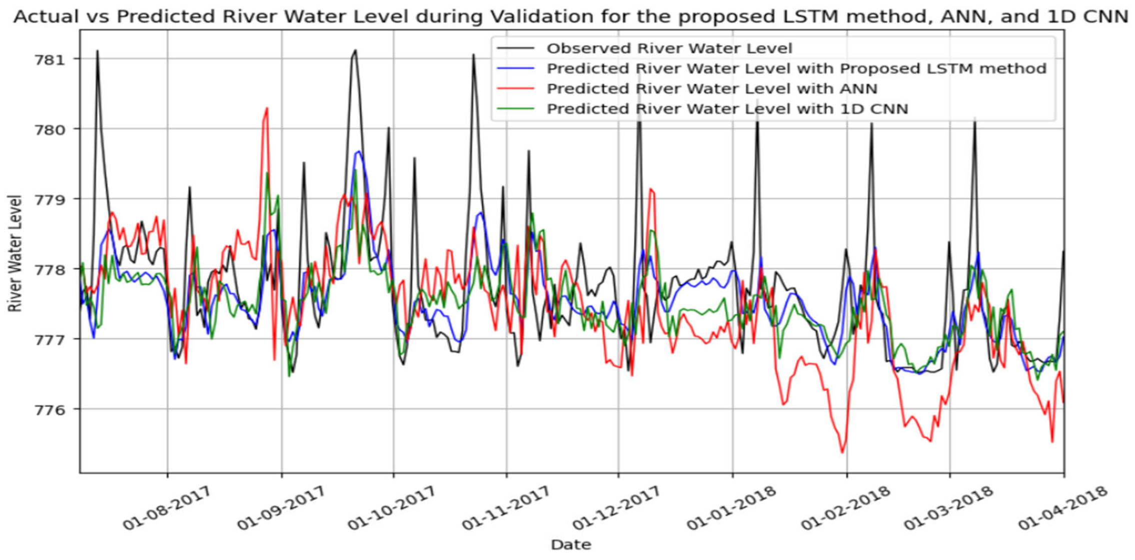
(h)Validation graph of 8-day lag



(l)Testing graph of 10-day lag

Figure 9. Figures (a) to (l) illustrate the actual and predicted river water levels for different lag periods of 3, 5, 7, and 10 days, using 1D CNN model with 16, 32, and 64 nodes.

(a)


(b)


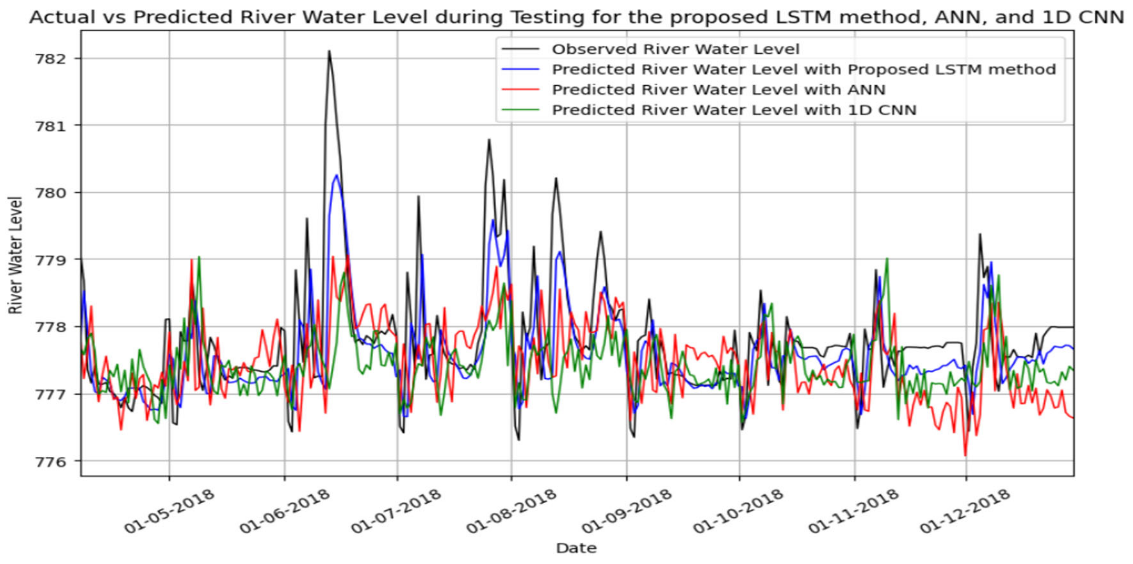
(c)


Figure 10. Figures (a), (b), and (c) illustrate the comparison of actual and predicted river water levels for the training, validation, and testing phase using the proposed LSTM model, ANN and 1D CNN.

Table 5. Comparison between the optimized configuration of the proposed KNN-LSTM method, ANN, and 1D CNN

Methods	Configuration	Data Split	NSE	MAPE	RMSE
Proposed KNN-LSTM Method	5-day lag, 32 Neurons	Training	0.9454	0.0869	1.0988
		Validation	0.9445	0.0941	1.1421
		Testing	0.9393	0.1076	1.1828
ANN	3-day lag, 64 Neurons	Training	0.8990	0.1367	1.4951
		Validation	0.8979	0.1421	1.5506
		Testing	0.8944	0.1449	1.5608
1D CNN	3-day lag, 16 filters	Training	0.9276	0.1054	1.2911
		Validation	0.9167	0.1239	1.3999
		Testing	0.9111	0.1327	1.4320

6. CONCLUSION

This study develops a flood forecasting model utilizing deep learning techniques, with a focus on predicting river water levels in Manipur. The model employs a multivariate, multiple-time-series LSTM architecture, which is specifically designed to capture the intricate temporal and spatial dynamics present in the dataset. To effectively address the issue of missing data, particularly in the target variable (river water levels), a KNN model is implemented for imputation, ensuring the robustness and reliability of the predictive model. The results of the comparative analysis demonstrate that the proposed KNN-LSTM model significantly outperforms conventional methods, including ANN and 1D CNN. This superior performance is evident across key evaluation metrics, such as NSE, MAPE, and RMSE. The findings underscore that the LSTM model, particularly when configured with a 5-day time lag and optimized network complexity, offers the most accurate and dependable predictions of river water levels. These results highlight the model's potential as a valuable tool for flood forecasting and water resource management in regions prone to hydrological variability.

Beyond just forecasting, the algorithm developed in this study also plays a crucial role in early warning systems. By accurately predicting river water levels, the model can trigger timely alerts, allowing authorities and communities to take preventive measures against potential floods. This capability is essential for minimizing the loss of lives, property, and infrastructure in flood-prone regions like Manipur.

This study underscores the potential of advanced deep learning models in enhancing flood forecasting and early warning systems, offering a valuable tool for disaster management in vulnerable areas. The findings also suggest that integrating robust imputation methods with deep learning models can significantly improve prediction quality, even in the presence of incomplete data. However, the model may exhibit reduced performance when test on datasets containing extreme values beyond the training range. The use of hourly data, instead of daily data, might increase its effectiveness for small river basins where rapid changes occur. Furthermore, producing accurate longer lead forecasts remains challenging due to the inherent uncertainties in hydrological processes.

Future work should focus on addressing these limitations by incorporating higher temporal resolution data, exploring additional environmental variables, and extending the application of the model to other regions with similar hydrological challenges.

ACKNOWLEDGEMENT

The authors would like to thank Directorate of Environment, Manipur, and Water Resource Department, Manipur, for providing daily rainfall, temperature, humidity and daily river water level data from six gauge stations.

REFERENCES

Adam, C. (2020). VigiFlood: Evaluating the Impact of a Change of Perspective on Flood Vigilance. *Journal of Integrated Disaster Risk Management, 10*(1), 69-103. https://doi.org/DOI10.5595/001c.17844

Bergamaschi, S., Beneventano, D., Guerra, F., & Orsini, M. (2011). Data Integration. In D. W. Embley, & B. Thalheim (Eds.), *Handbook of Conceptual Modeling* (pp. 441-476). Berlin: Springer.

Borrohou, S., Fissoune, R., Badir, H., & Tabaa, M. (2023). Data Cleaning in Machine Learning: Improving Real Life Decisions and Challenges. In J. Kacprzyk, M. Ezziyyani, & V. E. Balas (Eds.), *International Conference on Advanced Intelligent Systems for Sustainable Development* (pp. 627--638). Cham: Springer Nature Switzerland.

Cho, K., Merriënboer, B. v., Bahdanau, D., & Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. *Proceedings of {SSST}-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation* (pp. 103–111). Doha, Qatar: Association for Computational Linguistics. https://doi.org/10.3115/v1/W14-4012

Garavaglia, S. B., Dun, A. S., & HIll, B. M. (1998). A smart guide to dummy variables: Four applications and a macro. *Proceedings of the northeast SAS users group conference.* Retrieved from https://api.semanticscholar.org/CorpusID:8652875

Hamidreza, G. D., Reepal, S., Dimitrios, S., Yuhang, W., Boscovi, D., & John, S. (2019). Accurate prediction of streamflow using long short-term memory network: A case study in the Brazos river basin in Texas. *International Journal of Environmental Science and Development, 10*, 294--300. https://doi.org/10.18178/ijesd.2019.10.10.1190

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*, 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Houdt, G. V., Mosquera, C., & N´apoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review, 53*, 5929--5955. https://doi.org/10.1007/s10462-020-09838-1

Ghose, D. K. (2018). Measuring discharge using back-propagation neural network: a case study on brahmani river basin. In Embley, D., Thalheim, B. (Eds.), *Intelligent Engineering Informatics* (pp. 591–598). https://doi.org/doi:10.1007/978-981-10-7566-7_59

Kim, S., Matsumi, Y., Pan, S., & Mase, H. (2016). A real-time forecast model using artificial neural network for after-runner storm surges on the Tottori coast, Japan. *Ocean Engineering, 122*, 44-53. https://doi.org/https://doi.org/10.1016/j.oceaneng.2016.06.017

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, . Retrieved from https://arxiv.org/abs/1412.6980

Kulesh, M., Holschneider, M., & Kurennaya, K. (2008). Adaptive metrics in the nearest neighbours method. *Physica D: Nonlinear Phenomena, 237*, 283-291. https://doi.org/https://doi.org/10.1016/j.physd.2007.08.019

Kumar, D., Singh, A., Samui, P., & Jha, R. K. (2019). Forecasting monthly precipitation using sequential modelling. *Hydrological Sciences Journal, 64*, 690--700. https://doi.org/10.1080/02626667.2019.1595624

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*, 436–444. https://doi.org/10.1038/nature14539

Lohani, A. K., Goel, N., & Bhatia, K. (2014). Improving real time flood forecasting using fuzzy inference system. *Journal of Hydrology, 509*, 25-41. https://doi.org/https://doi.org/10.1016/j.jhydrol.2013.11.021

Maddox, R. A., Zhang, J., Gourley, J. J., & Howard, K. W. (2002). Weather Radar Coverage over the Contiguous United States. *Weather and Forecasting, 17*, 927 - 934. https://doi.org/10.1175/1520-0434(2002)017<0927:WRCOTC>2.0.CO;2

Maier, H. R., Jain, A., Dandy, G. C., & Sudheer, K. P. (2010). Methods used for the development of neural networks for the prediction of water resource variables in river systems: Current status and future directions. *Environmental Modelling and Software, 25(8)*, 891-909. https://doi.org/10.1016/j.envsoft.2010.02.003

Mosavi, Amir, Ozturk, P., & Chau, K.-w. (2018). Flood prediction using machine learning models: Literature review. *Water, 10*(11), 1536. doi:https://doi.org/10.3390/w10111536

Mulia, F. A., & Handayani, W. (2024). Assessment and Comparison of Community Resilience to Floods and Tsunamis in Padang, Indonesia. *Journal of Integrated Disaster Risk Management, 14*(1). https://doi.org/https://doi.org/10.5595/001c.115826

Ni, L., Wang, D., Singh, V. P., Wu, J., Wang, Y., Tao, Y., & Zhang, J. (2020). Streamflow and rainfall forecasting by two long short-term memory-based models. *Journal of Hydrology, 583*, 124296. https://doi.org/https://doi.org/10.1016/j.jhydrol.2019.124296

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence, 46(1-2)*, 77–105. https://doi.org/doi:https://doi.org/10.1016/0004-3702(90)90005-K

Ridzuan, M. R., Razali, J. R., Ju, S.-Y., & Rahman, N. A. (2024). That is Not My House? Household Renters' Flood Preparedness Intention in The East Coast Region of Malaysia. *Journal of Integrated Disaster Risk Management, 14*(1). https://doi.org/https://doi.org/10.5595/001c.92758

Rumelhart, D. E. (1987). Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations* (pp. 318-362). MIT Press.

Singh, T. B. (2022). Community Resilience and Chronic Flood in Imphal City. In A. Singh (Ed.), *International Handbook of Disaster Research* (pp. 1-13). Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-16-8800-3_18-1

Sophia, A., & Devi, M. S. (2020). ENVIRONMENTAL PROBLEMS OF MANIPUR. *The holistic approach to environment, 10*(4), 124 - 140. https://doi.org/https://doi.org/10.33765/thate.10.4.4

Weerakody, P. B., Wong, K. W., Wang, G., & Ela, W. (2021). A review of irregular time series data handling with gated recurrent neural networks. *Neurocomputing, 441*, 161-178. https://doi.org/https://doi.org/10.1016/j.neucom.2021.02.046

Yann, L., & Yoshua, B. (2002). Convolutional Networks for Images, Speech, and Time Series. In M. A. Arbib (Eds.), *Handbook of Brain Theory and Neural Networks.* (p. 3361). Cambridge: MIT Press.

Yu, Y., Zhang, H., & Singh, V. P. (2018). Forward prediction of runoff data in data-scarce basins with an improved ensemble empirical mode decomposition (eemd) model. *Water, , 10(4)*, 388. https://doi.org/doi:https://doi.org/10.3390/w10040388.

Zhang, J., Hou, G., Ma, B., & Hua, W. (2018). Operating characteristic information extraction of flood discharge structure based on complete ensemble empirical mode decomposition with adaptive noise and permutation entropy. *Journal of Vibration and Control, 24*, 5291-5301. https://doi.org/10.1177/1077546317750979

Zhongrun, X., Jun, Y., & Ibrahim, D. (2020). A Rainfall-Runoff Model With LSTM-Based Sequence-to-Sequence Learning. *Water Resources Research, 56*, e2019WR025326. https://doi.org/https://doi.org/10.1029/2019WR025326